Chapter 14

Communication Topology

14.1 System Topologies

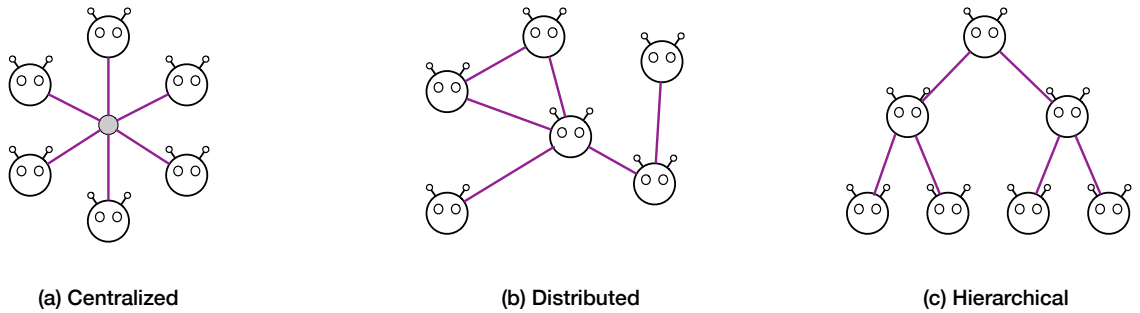

Figure 14.1: Different types of topological structure for multi-agent collaboration.

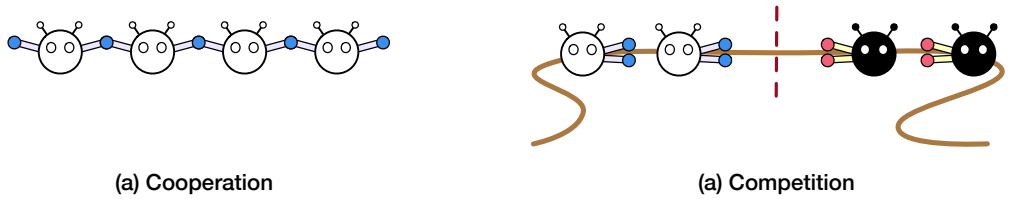

Figure 14.2: Collaborative and competitive agents.

This section examines the interaction typology in LLM-based multi-agent systems (MAS) and its impact on communication, collaboration, and task execution. We first analyze static topologies—where connectivity patterns are fixed by domain knowledge—and then explore dynamic (adaptive) topologies that adjust inter-agent connections based on performance metrics, workload variations, or strategic constraints. We conclude with a discussion of scalability challenges and trade-offs in balancing system cost, performance, and robustness, drawing on recent research in distributed processing, self-organization, and emergent collaborative behaviors.

14.1.1 Static Topologies

Static topologies are defined by predetermined structural patterns that remain largely unchanged during system execution. In these configurations, connections among agents—or between agents and a central coordinator—are established using fixed rules and heuristics, ensuring predictable communication flows and simplified coordination. Three canonical forms are typically considered: layered (hierarchical), decentralized, and centralized architectures.

Layered (Hierarchical) Structures Layered topologies arrange agents hierarchically, with high-level agents coordinating or supervising lower-level ones. This approach mirrors traditional management frameworks—such as Standard

Operating Procedures (SOP) or the Waterfall model—where tasks are decomposed into sequential, well-defined stages. For instance, the AutoAgents [1008] framework assigns roles (e.g., Planner, Agent Observer, and Plan Observer) to synthesize execution plans, while ChatDev [983] leverages hierarchical task decomposition to streamline software development [626, 921, 627]. Although hierarchical structures facilitate debugging, performance monitoring, and modularity, they can create bottlenecks when upper-tier agents are overloaded [1011]. Recent studies in storytelling [1012, 1013, 1014] and data science applications including data cleaning [1015, 1016], visualization [1017, 1018] and auto machine learning [1019, 1020], highlight the trade-off between consistency and the emergence of adaptive real-time behaviors.

Decentralized Structures In decentralized topologies, agents interact on a peer-to-peer basis without a central coordinator, forming networks that are often modeled as chains, rings, small-world, or random graphs [1021, 971]. This structure enhances fault tolerance since the failure of a single agent does not compromise the network. For example, [1022] show that distributing graph reasoning tasks among multiple agents enables scalability beyond the context length limits of individual LLMs. Additionally, [1023] propose decomposition strategies that allow an orchestrating LLM to delegate subtasks effectively. However, maintaining a coherent global state in decentralized systems necessitates sophisticated consensus and synchronization protocols.

Centralized Structures Centralized topologies rely on a master coordinator that gathers information and directs peripheral agents hierarchically. Such a setup allows for better control over handling resources and sharing a global view, such as with culture parks and Lyfe Agents [1024, 1025]. With additional agents, however, a bottleneck at the center node may occur, with increased communication overhead and susceptibility to failures. Current studies on coordinator-agent configurations [971] and research on ensuring autonomy for centralized configurations [1026] point out problems with scalability with consistency. While consistency is guaranteed for centralized architectures, there may not necessarily be flexibility for dynamic adaptation.

Briefly, static topologies have advantages of determinism and predefinition. With pre-defined structural patterns, these systems have predictable communication patterns and effective coordination among agents. Topologies of these structures are typically defined on structural knowledge or static rules, and, as such, they suit domains where workflow for the tasks is static, there are predefined roles, and system requirements are well defined. The second primary advantage is design, implementation, and maintenance ease. With structure predefined, design as well as execution procedures are made simpler, and, as a result, maintenance is a simpler process. Resource handling as well as modularization gets simpler due to well-defined, static structure.

However, static topologies themselves are nonflexible, grounded on pre-specified patterns of connectivity that do not respond to real-time changes. Well suited for a specific purpose at design time but entirely lacking flexibility for reacting to unforeseen challenges, including sudden agent breakdown, varying degrees of task complexity, and system goal modification, static topologies do not have real-time response flexibility potential. Real-time response inflexibility inhibits runtime system reconfiguration and decreases system effectiveness in dynamic settings where circumstances occur. Failure to self-organize and morph according to emerging conditions may equate to inefficiency as well as low system performance, particularly where dynamic or emergent settings are at hand.

14.1.2 Dynamic and Adaptive Topologies

While static topologies provide determinism and predictability—illustrated by static topologies such as hierarchical or centralized ones performing well with stable-task domains and well-defined roles—static topologies do not fit open-ended or novel domains. Real domains, from real-time collaborative plan, to dynamic social simulations, often demand that agents make changes on their patterns of interaction as work continues, available resources vary, or feedback from the environment is received. Such structural tension with adaptative malleability generates dynamic topologies, which, at runtime, recast inter-agent relationships as a response to feedback on performance, workload, or strategic constraints, striking a balance between consistency and responsiveness.

For example, DyLAN framework [725] supports inference-time agent selection through a two-step process: a forward-backward team optimization step with unsupervised Agent Importance Scores, followed by dynamic team reformulation at runtime. Similarly, OPTIMA [1027] optimizes inter-agent connectivity iteratively through a generate–rank–select–train framework, utilizing reward functions as a means for determining a balance among task quality, token efficiency, and readability, with communication actions further optimized through strategies such as Direct Preference optimization. The MAD framework [649] illustrates flexibility through a joint optimization among three prompt phases and structure, with dynamic role assignment (such as verifiers and debate participants) within pruned spaces for structure.

Topological control also becomes tractable through technological advancements. GPTSwarm [651] conceptualizes agents as computation graphs and uses evolutionary strategies and reinforcement learning for adjusting adjacency matrices for optimizing nodes based on feedback for the task. MACNET [1028] uses a directed acyclic graph architecture with supervisory instructors managing edges and executive assistants managing nodes for more complex coordination domains, facilitating adaptive communication through topological ordering and sensitive propagation of output. Application-specific versions also emphasize architecture diversity. Open-world environments have DAMCS [1029], which couples hierarchical knowledge graphs (A-KGMS) with structured communication schemes (S-CS) for cooperative planning as a function of messages passed based on context. AutoAgents [1030] leverages a dynamic drafting-execution pipeline with pre-defined agents jointly sketching out expert teams, a design that's highly effective for creative applications such as novel generation through parallel processing and internal supervision. Noticeably, small-world development within large-scale MACNET [1028] systems corresponds with graph reasoning ideas shown in [1022], where distributed architecture bypasses local limitations of LLM through structured collaboration. In terms of collaborative task solving, several paradigms have emerged that emphasize the role of dynamic topologies. These paradigms include search-based methodologies, LLM-based generation, and configurations utilizing external parameters.

Search-based Methods A number of works adopt search-based methodologies to iteratively optimize communication structures. For example, ADAS [741] employs a Meta Agent Search algorithm that iteratively generates and tests new agent designs within a code space, archiving superior configurations and thereby updating subsequent generation strategies. Similarly, Aflow [773] models each LLM call as a node in a graph and utilizes Monte Carlo Tree Search (MCTS) to dynamically extend and refine the workflow. Other frameworks, such as MAD [1031] and OPTIMA [1027], integrate iterative generate–rank–select–train paradigms that echo MCTS principles to balance task performance with efficiency.

LLM-based Methods Complementing search-based methods, several recent works leverage the generative capacity of LLMs to construct and adapt dynamic topologies. Dylan [725] introduces a temporal feed-forward network (T-FFN) model that treats each communication step as a network layer, using forward-backward propagation to compute Agent Importance Scores for dynamic team selection. In related work, DAMCS [1029], AutoAgents [1030], and TDAG [1032] dynamically generate specialized sub-agents or update hierarchical knowledge graphs, enabling cooperative planning and task decomposition. Further, frameworks such as AutoFlow [773] and Flow [1033] represent task workflows in natural language programs or activity vertex graphs (AOV), allowing continuous refinement through reinforcement learning signals. ScoreFlow [788] complements these approaches by applying gradient-based (loss-gradient) optimization to continuously reconfigure agent workflows.

External Parameters Given that fine-tuning LLM-based agents is often resource-intensive, a considerable number of researchers advocate configuring inter-agent topologies by training parameters independent of the LLM-agent. This approach is initiated by GPTSwarm [651], in which the inter-agent topologies are represented as a directed acyclic graph (DAG), with edge weights serving as the sole trainable component of the system. Further advancing this paradigm, AgentPrune provides a unified modeling framework from the spatial-temporal graph perspective for mainstream MAS, where communication redundancy, i.e., unnecessary edges, is identified and pruned through magnitude-based pruning. Follow-up works in this line of research include G-Safeguard [1034], which similarly trains GNN outside of the MAS to detect and eliminate malicious communication paths. Although these methods are parameter-efficient, their relatively small parameter space and low coupling with LLM-agents often result in performance limitations to some extent.

Discussion Dynamic topologies extend beyond task-solving and play a crucial role in simulating complex social interactions. As detailed in a recent survey [975], LLM-based agent models can evolve inter-agent links to capture real-time changes in autonomy, social behaviors, and environmental feedback across various domains, including cyber, physical, and mixed environments. Systems such as [50], OASIS [936] and ProjectSid [989] simulate dynamic social networks. [50] employs generative natural language memory retrieval to adjust social ties based on agents' experiences, while OASIS constructs a real-time social media environment with continuously updated user relationships and information flows. Project Sid [989] introduces the PIANO (Parallel Information Aggregation via Neural Orchestration) architecture, enabling over 1,000 autonomous AI agents to interact in real-time within a Minecraft environment, leading to the emergence of complex societal structures such as specialized roles, collective rule adherence, and cultural and religious transmission. Additionally, architectures like AgentScope-scability [1035] and Social Survey [975] support large-scale multi-agent simulations, enabling studies of cultural dissemination, collective decision-making, and emergent group dynamics in environments with hundreds or thousands of interacting agents. Additionally, dynamic topologies are also tailored to specific application domains such as medical and open-domain embodied AI. In the medical field, AI hospital [1036] and agent hospital [921] simulate real medical workflows, where iterative cycles of diagnosis, treatment, and feedback continuously reshape communication patterns among various roles, such as intern doctors,

patients, examiners, and supervising physicians. These frameworks dynamically adjust inter-agent communication to optimize collaboration and decision-making. Similarly, in open-domain and embodied AI applications, frameworks like IOA [933] support heterogeneous, cross-device agent interactions, facilitating dynamic team formation and task allocation in real-world scenarios.

Although the aforementioned dynamic multi-agent topologies have made substantial progress in performance metrics, they still face the following three limitations, which we believe should be the focal points for future research on dynamic topologies:

(1) Generalizability. Current MAS topologies are typically optimized for a single-task domain. For example, AFlow [773] is dedicated to search and optimization within math or code benchmarks, producing a fixed workflow that is difficult to adapt to new task domains. Other dynamic topologies, such as ADAS [741], GPTSWarm [651], and AgentPrune, face the same challenge. We argue that MAS should be capable of lifelong learning, wherein the system generalizes across different task domains with minimal resources (e.g., API calls, FLOPs, GPU hours).

(2) Resource Efficiency. Present dynamic topologies often tend to optimize for complex, resource-intensive structures. Their training processes are typically exorbitantly costly, as exemplified by ADAS [741], where training with GPT-3.5 incurs a cost of approximately $300 per session. Such expenses severely constrain their large-scale applicability in real-world scenarios. Future developments should focus on achieving better test-time topology optimization with significantly reduced costs.

(3) Inference Efficiency. As MaAS [787] has incisively observed, multi-agent topologies of excessive complexity, while capable of consistently delivering satisfactory performance, are lamentably deficient in *task adaptability*. That is to say, they are unable to dynamically allocate reasoning resources (i.e., tools, the number of agents, and reasoning steps) in response to the difficulty of a given task. Consequently, this may lead to a certain lack of efficiency in the inference process. Although MaAS has, to a certain extent, achieved task dynamism through the designed agentic supernet, their applicability and scalability in large-scale deployment still remain to be tested.

14.2 Scalability Considerations

Scalability is a critical challenge in LLM-based multi-agent systems (MAS), especially as the number of agents grows. In fully connected networks, the number of communication paths grows quadratically, leading to a communication explosion that increases token usage and computational costs [1037, 626]. Centralized and layered topologies can experience synchronization bottlenecks if supervisory nodes are inundated by messages, whereas decentralized networks—while more fault tolerant—necessitate complex consensus algorithms to achieve a coherent global state.

Recent work such as [1028] demonstrates that when multi-agent collaboration is structured as a directed acyclic graph (DAG), the system can scale efficiently to handle large graphs—up to 1,000 nodes or more—without significant performance degradation. Similarly [1022] shows that distributing graph reasoning tasks among many agents circumvents the limitations imposed by long textual inputs and context-length constraints. Moreover, studies on self-organized agents[1038] reveal that dynamic multiplication and task distribution allow the system to maintain a constant workload per agent while increasing overall processing capacity. Finally, the multi-dimensional taxonomy proposed by [1039] provides a valuable framework for analyzing trade-offs between agent autonomy and alignment, offering insights into how to balance centralized control with decentralized flexibility to optimize scalability.

In addition to these foundational studies, recent advances in practical multi-agent platform design further enrich the scalability discussion. For example, AgentScope [1035] offers a developer-centric platform that leverages an actor-based distributed framework to enable seamless migration between local and distributed deployments. Its unified workflow and automatic parallel optimization significantly reduce the communication overhead and synchronization challenges that typically emerge as agent numbers increase. By incorporating fault-tolerance mechanisms and intelligent message filtering, AgentScope illustrates how system-level supports can be designed to maintain performance even in dynamic and heterogeneous deployment environments.

Another complementary approach is presented in Project Sid [989], which explores scalability within the realm of simulating agent civilizations. Here, the focus shifts from isolated task solving to the simulation of complex societal dynamics. The proposed PIANO (Parallel Information Aggregation via Neural Orchestration) architecture allows agents to operate concurrently by decoupling slower cognitive processes from rapid reactive modules. A dedicated cognitive controller is introduced to ensure coherence among multiple parallel outputs. This design not only enables scalability from small groups to simulations involving over a thousand agents but also effectively addresses the inherent coordination challenges arising from high-frequency interactions.

Taking scalability to an even larger scale, AgentSociety [1040] demonstrates a comprehensive framework for simulating realistic social environments with up to 10,000 agents. By integrating LLM-driven social generative agents within a realistic urban, social, and economic setting, AgentSociety employs distributed computing and a high-performance messaging system (e.g., MQTT) to support millions of daily interactions. This platform exemplifies how emerging hybrid architectures can support macro-level phenomena—such as economic market dynamics, opinion diffusion, and urban planning simulations—by effectively managing the trade-offs between communication cost, coordination overhead, and emergent behavior fidelity.

Despite the theoretical advantages of scaling up agent populations, it is imperative to question whether pursuit of large-scale agent deployments is inherently valuable for all task-solving scenarios. Although the total computational capacity scales with the number of agents, when memory overhead and inter-agent communication costs are factored in, the marginal utility of adding additional agents may demonstrate diminishing returns. This phenomenon arises from the fundamental constraint that, while the overall workload is the product of individual task complexity and the degree of labor division, coordination costs tend to increase super-linearly with agent count. Therefore, for many bounded problem domains, there is likely an optimal agent population size beyond which performance plateaus—or even deteriorates—due to excessive coordination overhead.

Conversely, in simulation scenarios where the objective is to model complex social dynamics, emergent behaviors, or large-scale collective intelligence, scaling to numerous agents becomes not merely beneficial but essential. In these contexts, the research focus shifts from optimizing computational efficiency for task solving to accurately reproducing or predicting macro-level patterns emerging from micro-level agent interactions. Such simulations—covering domains like economic market behavior, social network evolution, and urban infrastructure planning—often require the computational overhead of managing vast agent populations in order to capture realistic population-level phenomena.

Hybrid architectures that combine centralized oversight with decentralized sub-teams offer a promising solution to these scalability challenges [921, 918]. In these designs, supervisory agents handle global objectives and coordination, while worker agents focus on executing specific subtasks. This hierarchical organization helps to mitigate information overload at any single node and allows for dynamic adjustment of agent team sizes based on task demands, thereby optimizing resource utilization. Furthermore, advanced techniques such as graph search algorithms, reinforcement learning-based updates, and evolutionary methods are critical for iteratively refining the network structure as the system scales. Intelligent message filtering, prioritization, and aggregation mechanisms can significantly reduce communication overhead without sacrificing the quality of inter-agent collaboration. In addition, asynchronous communication protocols and partial knowledge sharing strategies show promise in minimizing coordination bottlenecks while maintaining sufficient global awareness among agents.

Concluding Remarks on Scalability Overall, the study of system topology and scalability in LLM-based MAS reveals a spectrum of design choices—from static configurations that offer simplicity and predictability to dynamic architectures that provide flexibility and adaptability. While foundational works (e.g., [1028], [1038]) emphasize scalable graph structures and self-organizing principles, the practical advances demonstrated by AgentScope, Project Sid, and AgentSociety illustrate how integrated distributed frameworks, concurrent processing, and realistic environment simulations can collectively address the challenges of scaling multi-agent systems. The context-dependent nature of scalability requirements—contrasting between task-solving and simulation scenarios—highlights the importance of purpose-specific design in multi-agent architectures. As research continues to evolve, the development of more sophisticated adaptive algorithms, distributed architectures, and multi-dimensional evaluation frameworks will be essential for advancing the scalability and practical viability of LLM-based multi-agent systems.